JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 93, No. 3, pp. 619–634, JUNE 1997

Piecewise-Linear Pathways to the Optimal Solution Set in Linear Programming[1]

M. Ç. PINAR[2]

Communicated by P. Tseng

Abstract. This paper takes a fresh look at the application of quadratic penalty functions to linear programming. Recently, Madsen et al. (Ref. 1) described a continuation algorithm for linear programming based on smoothing a dual l_1-formulation of a linear program with unit bounds. The present paper is prompted by the observation that this is equivalent to applying a quadratic penalty function to the dual of a linear program in standard canonical form, in the sense that both approaches generate continuous, piecewise-linear paths leading to the optimal solution set. These paths lead to new characterizations of optimal solutions in linear programming. An important product of this analysis is a finite penalty algorithm for linear programming closely related to the least-norm algorithm of Mangasarian (Ref. 2) and to the continuation algorithm of Madsen et al. (Ref. 1). The algorithm is implemented, and promising numerical results are given.

Key Words. Quadratic penalty functions, linear programming, piecewise-linear path-following algorithms, characterization of optimal solutions, finiteness.

1. Introduction

In a recent paper (Ref. 1) Madsen et al. gave a new finite algorithm to solve the problem

$$(\text{AL1}) \quad \min_x G(x) \equiv \|E^T x - d\|_1 + f^T x, \tag{1}$$

[1]This research was supported by Grant No. 11-0505 from the Danish Natural Science Research Council SNF. The author is indebted to K. Madsen, H. B. Nielsen, and V. A. Barker for their support. The careful work of two anonymous referees is also gratefully acknowledged.

[2]Assistant Professor, Department of Industrial Engineering, Bilkent University, Bilkent, Ankara, Turkey.

0022-3239/97/0600-0619$12.50/0

where E is $n \times m$, $d \in \mathfrak{R}^m$, and $f \in \mathfrak{R}^n$. The case where $f \equiv 0$ is precisely the linear l_1-estimation problem (Ref. 3). A smooth approximation of (AL1) was considered by Madsen et al. (Ref. 1):

$$\text{(SAL1)} \quad \min_x G_\gamma(x) \equiv \sum_{i=1}^{n} \rho(z_i(x)) + f^T x, \tag{2}$$

where

$$z_i(x) = e_i^T x - d_i, \tag{3a}$$

$$\rho(z_i) = \begin{cases} (1/2\gamma) z_i^2, & \text{if } |z_i| \le \gamma, \\ |z_i| - \gamma/2, & \text{otherwise.} \end{cases} \tag{3b}$$

The function ρ is known as the Huber function in robust regression; see Ref. 4. A finite Newton type algorithm was used by Madsen and Nielsen (Ref. 5) to solve (SAL1). Later, Madsen et al. (Ref. 1) showed that the set of minimizers of G_γ define continuous, piecewise-linear paths as a function of γ leading to the optimal solution set of (AL1). They established that solutions to (AL1) and its dual (which is a linear program with unit upper and lower bounds) can be found for a sufficiently small positive value of the parameter γ. By exploiting this property, a finite continuation algorithm to solve linear programs was given in the same reference with encouraging initial numerical results. Now, the present paper is prompted by the observation that the ideas summarized in the above paragraphs can be used to solve the linear programming problem using quadratic penalty functions.

Consider the primal linear programming problem

$$\begin{aligned} \text{(P)} \quad & \min \quad c^T x, \\ & \text{s.t.} \quad Ax = b, \\ & \qquad x \ge 0, \end{aligned}$$

where $x \in \mathfrak{R}^n$, A is an $m \times n$ matrix, $b \in \mathfrak{R}^m$, $c \in \mathfrak{R}^n$. The dual problem to (P) is given by

$$\begin{aligned} \text{(D)} \quad & \max \quad -b^T y, \\ & \text{s.t.} \quad A^T y + c \ge 0, \end{aligned}$$

where $y \in \mathfrak{R}^m$. We define the dual penalty functional

$$H(y, t) \equiv t b^T y + (1/2) r^T(y) W r(y), \tag{4}$$

where t is a positive scalar,

$$r(y) \equiv A^T y + c,$$

and W is a diagonal $m \times m$ matrix with entries:

$$W_{ii}(y) = \begin{cases} 1, & \text{if } r_i(y) < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

In this paper, we study the unconstrained problem

$$\text{(CD)} \quad \min_y H(y, t), \tag{6}$$

for decreasing positive values of t.

All the results and the algorithm of the present paper are obtained by combining the continuation viewpoint (Ref. 1) and a well-known duality and perturbation result in linear programming (Refs. 2, 6–10) to study the quadratic penalty problem (CD). This result states that, for t sufficiently small, a dual optimal point to the penalty problem (CD) is a solution of (P) with the least 2-norm. Using this result, we establish the piecewise linear behavior of the solution set of (CD) as t is decreased, and give novel characterizations of the solution set of (D). The main contribution of the paper is the design of a finite algorithm as a modification of the least-norm algorithm of Mangasarian (Ref. 2) using the ideas of Madsen et al. (Ref. 1). However, the algorithm of the present paper differs from the continuation algorithm of Ref. 1 in the strategy for reducing the parameter t and the stopping criteria. The algorithm of Mangasarian (Ref. 2) computes a least-norm solution of linear programs based on the above perturbation result and using a SOR (successive overrelaxation) algorithm. Our algorithm differs from this algorithm in two important aspects: (i) we use a modified Newton algorithm to compute a point on the paths; (ii) we use a new, effective reduction strategy for t that exploits the piecewise-linear nature of solution paths and yields primal–dual solutions in a finite calculation.

We give computational results of our algorithm using a small subset of the Netlib test set. The results indicate that the new dual penalty algorithm exhibits a performance comparable to the algorithm of Ref. 1 and to the dense linear programming system LSSOL from Stanford University Systems Optimization Laboratory (Ref. 11) on the test set. For more details on penalty and active set methods for linear programming, see Refs. 6, 12–16.

While the present paper was under review, we have developed an application of the quadratic penalty functions to linear programs with mixed inequality and equality constraints. This is both theoretically and computationally a nontrivial extension of the present paper and is described in detail in Pinar (Ref. 17). This also relates our research to early work by Chebotarev in the Russian literature (Ref. 14).

2. Pathways to Optimal Solutions

Let y_t denote a solution to (CD) for some $t>0$. It is well known (see, e.g., Ref. 18) that every limit point of the sequence $\{y_t\}$ solves (D). Now, define a binary vector $s\in\mathfrak{R}^n$ such that

$$s_i(y)=\begin{cases}1, & \text{if } r_i(y)<0,\\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

Hence,

$$W(y)=\operatorname{diag}(s_1(y),\ldots,s_m(y)).$$

In the sequel, we drop the argument y when the meaning is clear from context. We denote by Y the set of optimal solutions to (D). We assume throughout the paper that A has rank m.

2.1. Structure of the Solution Set of (CD). In this section, we briefly examine the properties and structure of the set of minimizers of H. The results of this section follow immediately using some results of Refs. 8 and 9.

It is evident that H is composed of a finite number of quadratic functions. In each domain $D\subseteq\mathfrak{R}^m$ where $s(y)$ is constant, H is equal to a specific quadratic function as seen from its definition. These domains are separated by the following union of hyperplanes:

$$B=\{y\in\mathfrak{R}^m\mid\exists i\colon r_i(y)=0\}. \tag{8}$$

A binary vector s is feasible at y if,

$$\forall\epsilon>0,\qquad \exists z\in\mathfrak{R}^m\setminus B\colon \|y-z\|<\epsilon\wedge s=s(z).$$

If s is a feasible binary vector, H is given by a specific quadratic function on the subset

$$\mathscr{C}_s=\operatorname{cl}\{z\in\mathfrak{R}^m\mid s(z)=s\}. \tag{9}$$

We also call $\mathscr{C}_s$ a Q-subset of $\mathfrak{R}^m$. The gradient of the function H is given by

$$H'(y,t)=AWr(y)+tb, \tag{10}$$

where W is the diagonal matrix obtained from a feasible binary vector s at y. For $y\in\mathfrak{R}^m\setminus B$, the Hessian of H exists and is given by

$$H''(y,t)=AWA^T. \tag{11}$$

We also denote by U_t the set of minimizers of $H(y,t)$.

It is well known (see for instance Refs. 2, 6–10) that the dual of the quadratic penalty problem (CD) is given as

$$\begin{aligned} \text{(PB)} \quad \min \quad & c^T x + (t/2) x^T x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

Let x_t denote the unique optimal solution to the above perturbed program, and let z_+ denote a vector whose ith component is $\max\{z_i, 0\}$. It is shown in Lemma 6 of Ref. 9 that

$$tx_t = -(A^T z_t + c)_+, \tag{12}$$

for any $z_t \in U_t$. That is, the right-hand side is constant regardless of the choice of minimizer z_t. Hence, we have the following characterizations of the solution set of (CD) analogously to the solution set of (SAL1).

Lemma 2.1. $s(y_t)$ is constant for $y_t \in U_t$. Furthermore, $r_i(y_t)$ is constant for $y_t \in U_t$ if $s_i = 1$.

Following the lemma, we let

$$s(U_t) = s(y_t), \qquad y_t \in U_t,$$

as the sign vector corresponding to the solution set. Now, we can use Lemma 2.1 and the linearity of the problem to characterize the solution set U_t.

Corollary 2.1. U_t is a convex set which is contained in a Q-subset $\mathscr{C}_s$, where $s = s(U_t)$.

Corollary 2.2. Let $y_t \in U_t$ and $s = s(U_t)$. Let $\mathscr{N}_s$ be the orthogonal complement of $\mathscr{V}_s = \text{span}\{a_i^T | s_i = 1\}$. Then,

$$U_t = (y_t + \mathscr{N}_s) \cap \mathscr{C}_s.$$

2.2. Characterization of Optimal Solutions through Dual Paths. In this section, we analyze the behavior of the solution set U_t as t approaches zero. The main result of this section is given in Theorem 2.2.

Assume that $y_t \in U_t$ and $s = s(U_t)$, with $W = \text{diag}(s_1, \ldots, s_n)$. Then, y_t satisfies the following identity:

$$(AWA^T) y_t = -AWc - tb. \tag{13}$$

It follows from (13) that the following linear system is consistent:

$$(AWA^T)d=b. \tag{14}$$

Now, let x_t denote the unique optimal solution to (PB). Then, we have from Li and Swetits (Ref. 9) that

$$x_t=-W(A^Ty_t+c)/t, \tag{15}$$

for any $y_t\in U_t$. From Theorem 2.1 of Mangasarian (Ref. 2), we have that, for small enough t, for $t\in(0, t_0]$ say, x_t is constant: it is the least 2-norm solution of (P). This implies that

$$x_t=-W(A^Ty_{t-\delta}+c)/(t-\delta), \tag{16}$$

for $0\le\delta<t$. Let

$$y_{t-\delta}=y_t+\delta h.$$

Then, one can rewrite the above as

$$x_t=-W(A^T(y_t+\delta h)+c)/(t-\delta). \tag{17}$$

Combining (15) with (17), we have

$$x_t=WA^Th.$$

Multiplying both sides by A, and since $Ax_t=b$, we get

$$AWA^Th=b.$$

Hence, using Corollary 2.2, we have the following theorem.

Theorem 2.1. There exists $t_0>0$ such that $s=s(U_t)$ is constant for $0<t\le t_0$. Furthermore, let d be a solution to (14) such that $s(y_t+\delta d)=s$ for $0\le\delta<t\le t_0$. Then,

$$U_{t-\delta}=(y_t+\delta d+\mathcal{N}_s)\cap\mathcal{C}_s, \qquad \text{for } 0\le\delta<t\le t_0.$$

In general, it follows from the linearity of the problem that the set of minimizers U_t has a piecewise-linear structure. Combining this with Corollary 2.2, we have the following corollary.

Corollary 2.3. Let $y_t \in U_t$ and let $s = s(U_t)$. Let d solve (14). If $s(y_t + \epsilon d) = s$ for $\epsilon > 0$, then $s(y_t + \delta d) = s$ and

$$U_{t-\delta} = (y_t + \delta d + \mathcal{N}_s) \cap \mathcal{C}_s, \qquad 0 \leq \delta \leq \epsilon. \tag{18}$$

Since the number of distinct binary vectors is finite, we also have the following corollary.

Corollary 2.4. $s(U_t)$ is a piecewise constant function of $t > 0$.

Combining the study of the piecewise-linear paths generated by the minimizers of H with Theorem 2.1 cf Mangasarian (Ref. 2), Theorem 2.2 below gives a new characterization of the optimal solution set Y of (D).

Theorem 2.2. Let $0 < t \leq t_0$, where t_0 is given in Theorem 2.1, and let $s = s(U_t)$. Let $y_t \in U_t$, and let d solve (14). Then, $U_0 \equiv Y$, where

$$U_0 = (y_t + td + \mathcal{N}_s) \cap \mathcal{D}_s, \tag{19}$$

with

$$\mathcal{D}_s \equiv \{y \mid a_i^T y + c_i \geq 0, \forall i \in \sigma_s\},$$

$$\sigma_s = \{i \mid 1 \leq i \leq n \wedge s_i = 0\}.$$

Proof. First, $x_t = -Wr(y_t)/t$ is an optimal solution to (P) with least norm by Theorem 2.1 of Ref. 2. Let $y \in Y$. By the complementary slackness theorem, $Wr(y) = 0$. Now, let $y_t \in U_t$. Then using (13), we have

$$\begin{aligned} AWA^T(y - y_t) &= AWA^T y + AWc + tb \\ &= AWr(y) + tb \\ &= tb. \end{aligned} \tag{20}$$

Hence, there exists a solution d to (14) such that $y = y_t + td$. Since any solution $\tilde{d}$ to (14) can be expressed as $\tilde{d} = d + \eta$, where $\eta \in \mathcal{N}(AWA^T)$,

$$Wr(y_t + t\tilde{d}) = 0,$$

for any solution d to (14). Now, assume $y_0 \in U_0$. Then, there exists a solution d_0 to (14) such that $y_0 = y_t + td_0$. Therefore, we have

$$Wr(y_0) = 0. \tag{21}$$

Now, y_0 is complementary to x_t and is feasible in (D) by definition of U_0 and (21). Therefore, y_0 is an optimal solution to (D). Since this holds for any $y_0 \in U_0$, $U_0 \subseteq Y$.

If Y is a singleton, the proof is complete. Therefore, assume otherwise. Let $y \in Y$. Using (20) again, it follows that $(y - y_t)/t$ solves (14) for any $y_t \in U_t$. Therefore, we have shown that $y \in y_t + td + \mathcal{N}_s$. Now, by virtue of feasibility, $y \in \mathcal{D}_s$ for any $y \in Y$. Hence, $y \in U_0$. □

3. Dual Penalty Algorithm

We now construct a penalty algorithm for linear programming similar to the finite continuation algorithm of Madsen et al. (Ref. 1) and to the least-norm algorithm of Mangasarian (Ref. 2). This algorithm is different from the continuation algorithm in two important aspects: (i) the choice of the reduction strategy for t; (ii) the stopping criteria. These differences make the algorithm a finite variant of the algorithm of Mangasarian. The algorithm of Mangasarian consists of solving the dual to (PB) using a SOR iterative scheme to compute a least-norm solution to (P). Our algorithm computes both a least-norm solution to (P) and an optimal solution to (D) in a finite calculation using the piecewise linear dependence of unconstrained minimizers on t. In contrast to the algorithm of Mangasarian, we use a finite Newton-type method to solve (CD), which is more accurate than the SOR method. On the other hand, the algorithm of Mangasarian is very simple, since it does not involve any matrix factorizations.

The algorithm is initiated by choosing values t^0 and y^0. A minimizer y_t^0 of H using t^0 is found. The general step $k+1$ for $k \geq 0$ is given below.

Step $k+1$. Let y_t^k denote a minimizer of H for t^k. Choose t^{k+1} and a starting point for subsequent iterations based on y_t^k and compute a minimizer y_t^{k+1} of H.

The algorithm stops when the duality gap is zero (within roundoff) and both primal and dual feasibility are observed. Otherwise, t is decreased according to some criteria; see Section 3.2. To complete the description, we need an algorithm to compute a minimizer of H. Such an algorithm is adapted from the Newton algorithm of Madsen and Nielsen (Ref. 5) for robust linear regression using Huber functions; see Section 3.1.

Now, we define an extended binary vector $\bar{s}$ such that

$$\bar{s}_i(y) = \begin{cases} 1, & \text{if } r_i(y) \leq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

We denote by $\bar{W}$ the diagonal matrix derived from $\bar{s}$. We also define the following active set of indices:

$$I(y) \equiv \{i \mid 1 \leq i \leq n \wedge \bar{s}_i(y) = 1\}. \tag{23}$$

This extended definition of binary vector is used in the detailed description of the algorithm and in proving its finite convergence.

3.1. Computing an Unconstrained Minimizer. Let y be the current iterate with $\bar{W}$ the diagonal matrix derived from $\bar{s}(y)$. A search direction h is computed by solving the Newton system

$$(A\bar{W}A^T)h = -A\bar{W}r(y) - tb. \tag{24}$$

For ease of notation, let

$$C \equiv A\bar{W}A^T, \qquad g \equiv -A\bar{W}r(y) - tb.$$

Furthermore, let $\mathcal{N}(C)$ denote the null space of C. If C has full rank, then h is the solution to (24). Otherwise, if the system of equations (24) is consistent, a minimum-norm solution is computed. If the system is inconsistent, the projection of g on $\mathcal{N}(C)$ is computed. These choices are motivated and justified in Madsen and Nielsen (Ref. 5). The next iterate is found through a line search aiming for a zero of the directional derivative. This procedure is computationally cheap as a result of the piecewise-linear nature of H'. It can be shown that the iteration is finite; i.e., after a finite number of iterations, we have that $y+h$ is a minimizer of H; see Ref. 5 for details.

3.2. Reducing the Penalty Parameter. Let y_t be a minimizer of $H(y, t)$ for some $t>0$ and $\bar{s}=\bar{s}(y_t)$. Consider the system

$$(A\bar{W}A^T)d = b. \tag{25}$$

Let d be a solution to (25). We distinguish between two cases.

Case 1. The duality gap $c^Tx + b^T(y_t + td)$ is zero, but $y_t + td$ is infeasible in (D); i.e., there exists j such that $r_j(y_t + td) < 0$. In this case, we reduce t as follows. Let $\phi \equiv \{\alpha_k, k = 1, 2, \ldots, q\}$ be the set of positive kink points where the components of $r(y_t + td)$ change sign, i.e., the set

$$\phi = \{0 < \alpha < 1 \mid \exists i \in J \mid r_i(y_t) + \alpha_i t a_i^T d = 0\},$$

where

$$J = \{i \mid 1 \le i \le n \text{ and } a_i^T d \ne 0\}.$$

If ϕ is nonempty, we choose $\alpha^* = \min_k \alpha_k$ and we let

$$t_{\text{next}} \equiv (1 - \alpha^*)t, \qquad y_{t_{\text{next}}} \equiv y_t + \alpha^* td.$$

Otherwise, we set

$$t_{\text{next}} \equiv 0.9t, \qquad y_{t_{\text{next}}} \equiv y_t + 0.9td.$$

Notice that

$$t_{\text{next}} = \beta t,$$

where either $\beta \leq 0.9$ or $\beta = 1 - \alpha^*$, with $0 < \alpha^* < 1$, so that $t_{\text{next}} < t$. In both cases, $y_{t_{\text{next}}}$ is used as the starting point of the modified Newton algorithm of Section 3.1 with the reduced value of t.

Case 2. The duality gap is not zero. In this case, we reduce t as follows. Let $c((1-\epsilon)t)$ denote the number of changes in the active set from $I(y_t)$ to $I(y_t + \epsilon t d)$. We use bisection to find a value $\bar{\epsilon}$ of ϵ such that $c((1-\bar{\epsilon})t) \approx 1/2c(t)$, and use

$$t_{\text{next}} = (1-\bar{\epsilon})t, \qquad y_{t_{\text{next}}} \equiv y_t + \bar{\epsilon} t d.$$

For robustness, we search only in the interval $[0.1t, t)$ so that $t_{\text{next}} \leq 0.9t$.

3.3. Finite Convergence. Let $\mathcal{S}(U_t)$ denote the set of all distinct binary vectors corresponding to the elements of U_t. That is, for any $y_t \in U_t$, $\bar{s}(y_t) \in \mathcal{S}(U_t)$.

The following results are obtained as consequences of the linearity of the problem, Lemma 2.1, and the finiteness of the extended binary vectors.

Lemma 3.1. If $\mathcal{S}(U_{t_1}) = \mathcal{S}(U_{t_2})$, where $0 < t_2 < t_1$, then $\mathcal{S}(U_t) = \mathcal{S}(U_{t_1}) = \mathcal{S}(U_{t_2})$ for $t_2 \leq t \leq t_1$.

Theorem 3.1. There exists $\bar{t}$ such that $\mathcal{S}(U_t)$ is constant for $t \in (0, \bar{t})$, where $0 < \bar{t} \leq t_0$.

The following theorem is crucial for the finite convergence analysis. Since the proof is lengthy, we refer the reader to the more general analysis of Ref. 17.

Theorem 3.2. Let $t \in (0, \bar{t})$ and $y_t \in U_t$, with $\bar{s} = \bar{s}(y_t)$ and $x^* \equiv -(1/t)\bar{W}r(y_t)$. Then,

$$\bar{W}r(y_t + td) = 0, \tag{26}$$

$$b^T(y_t + td) + c^T x^* = 0, \tag{27}$$

for any solution d to (25). Furthermore, if d is unique, $y_t + td$ solves (D).

Lemma 3.2. Assume $t \in (0, \bar{t})$. Let $y \in U_t$, with $\bar{s} = \bar{s}(y)$. Let d solve (25), and let y_{next} be generated by one iteration of the penalty algorithm. Then, either

$$y_{\text{next}} \equiv y + td \in Y,$$

and the algorithm stops, or

$$y_{\text{next}} \equiv y + \alpha^* td \in U_{t_{\text{next}}},$$

$$t_{\text{next}} = (1 - \alpha^*)t,$$

where α^* is as defined in Case 1 of the reduction procedure, and $I(y_{\text{next}})$ is an extension of $I(y)$.

Proof. Let

$$x = -(1/t)\bar{W}r(y).$$

Clearly,

$$c^T x + b^T(y + td) = 0$$

from Theorem 3.2. Hence, we are in Case 1 of the reduction procedure of Section 3.2. If $r(y + td) \geq 0$, then $y_{\text{next}} \equiv y + td$ is a solution to (D) by Theorem 3.2, and the algorithm stops. Otherwise, Theorem 3.2 implies that $I(y) \subseteq I(y + td)$. Hence, using the definition of α^*, we have

$$I(y + \alpha td) = I(y), \qquad \text{for } \alpha \in [0, \alpha^*).$$

Since there exists $j \in \{1, \ldots, n\} \setminus I(y)$ such that $r_j(y + \alpha^* td) = 0$, $I(y + \alpha^* td)$ is an extension of $I(y)$. Furthermore, $y + \alpha^* td \in \mathscr{C}_{\bar{s}}$. Therefore, using the continuity of the gradient H', (10), and the definition of d, have

$$H'(y, t) = H'(y + \alpha^* td, (1 - \alpha^*)t) = 0.$$

Thus, y_{next} minimizes $H(y, (1 - \alpha^*)t)$. □

Let $y \in U_t$ for some $t > 0$. Unless the stopping criteria are met and the algorithm stops with a primal–dual optimal pair, t is reduced by a nonzero factor ($t_{\text{next}} = \beta t$) as discussed in Section 3.2. Hence, $\{t_k\}$ is a strictly decreasing sequence converging to 0. Since the modified Newton iteration of Section 3.1 is a finite process, t reaches the range $(0, \bar{t})$, where $\bar{t}$ is as defined in Theorem 3.1, in a finite number of iterations unless the algorithm stops. Now, assume that $t \in (0, \bar{t})$. From Lemma 3.2, either the algorithm terminates or the active set I is expanded. Repeating this argument, in a finite number of iterations the matrix $A\bar{W}A^T$ will finally have rank m, since A has rank

m. When $A\bar{W}A^T$ has full rank, the solution d to the system (25) is unique, and $y_{\text{next}} = y + td$ solves (D) by Theorem 3.2. Therefore, we have proved the following theorem.

Theorem 3.3. The dual penalty algorithm defined in Section 3 terminates in a finite number of iterations with a primal–dual optimal pair.

4. Implementation and Numerical Results

In this section, we report our numerical experience with a Fortran 77 implementation of the dual algorithm, which does not exploit sparsity. The purpose of the experiments is to test the viability of the algorithm in solving problems of practical interest, and to compare it with the code LPASL1 of Madsen et al. with the package LSSOL from Stanford Systems Optimization Library. LSSOL is a Fortran 77 package for constrained linear least-square problems, linear programming, and convex quadratic programming; see Ref. 11. It does not exploit sparsity. Hence, it provides a fair comparison to our numerical results. We refer to the implementation of the penalty algorithm as LPPEN.

4.1. Numerical Linear Algebra. The major effort in the Newton algorithm of Section 3.1 is spent in solving the systems (24). We use the AAFAC package of Nielsen (Ref. 18) to perform this. The solution is obtained via an LDL^T factorization of the matrix $C_k = A\bar{W}A^T$. Let us recall that $\bar{W}_{ii} = 1$ for the columns of A corresponding to indices in the active set I as defined in (23). Based on this observation, D and L are computed directly from the active columns of A, i.e., without squaring the condition number as would be the case if C_k was first computed. It is essential for the efficiency of the penalty algorithm to observe that, normally, only a few entries of the diagonal matrix $\bar{W}$ change between two consecutive iterations. This implies that the factorization of C_k can be obtained by relatively few updates and downdates of the LDL^T factorization of C_{k-1}. Therefore, the computational cost of a typical iteration step is $O(m^2)$. Occasionally, a refactorization is performed. Details are given in Ref. 18.

For the solution of (25), the matrix $A\bar{W}A^T$ is identical to the last C_k used in the Newton iteration. Thus, the LDL^T factorization is available. In order to enhance accuracy, we use one step of the iterative refinement when solving (25). All the results reported in this study were obtained under identical algorithmic choices without fine tuning LPPEN for any particular problem.

Table 1. Solution statistics of LPPEN on the test set.

Problem name	Iter	Refac	Reduc	f^*	t^*	CPU
Afiro	24	2	9	$-4.647531429 \times 10^{2}$	6.07×10^{-4}	1.09
Sc50b	17	1	4	$-7.000000000 \times 10^{1}$	7.62×10^{-5}	1.59
Sc50a	25	2	8	$-6.457507706 \times 10^{1}$	7.71×10^{-5}	1.70
Sc105	37	3	7	$-5.220206121 \times 10^{1}$	8.11×10^{-6}	6.37
Adlittle	68	8	7	$2.254949632 \times 10^{5}$	3.28×10^{-4}	5.60
Scagr7	77	5	14	$-2.331389824 \times 10^{6}$	4.94×10^{-7}	17.36
Stocfor1	88	5	15	$-4.113197622 \times 10^{4}$	3.48×10^{-5}	16.20
Blend	63	8	13	$-3.081214985 \times 10^{1}$	7.02×10^{-4}	7.80
Sc205	61	5	14	$-5.220206121 \times 10^{1}$	6.17×10^{-7}	31.26
Share2b	99	5	24	$-4.157322407 \times 10^{2}$	2.89×10^{-5}	12.19
Plate	21	3	3	$2.400000000 \times 10^{1}$	1.84×10^{-3}	2.69

To initiate the algorithm, we choose a starting point y^0 and t^0 as follows. We let y^0 be a solution to

$$(AA^T)y = -Ac - \kappa b,$$

where $\kappa \in (0, 1]$. Then, t^0 is chosen using the following formula:

$$t^0 = \text{MINRES}(m, r^0) * \beta,$$

where $\beta \in (0, 1]$, $r^0 = A^T y^0 + c$, and MINRES is a function that returns the mth smallest element of r^0.

4.2. Test Results. We consider ten problems from the Netlib collection. We also used a test problem from a civil engineering application at the Technical University of Denmark, referred to as Plate. The characteristics of the test problems can be found in Ref. 1. We do not use any preprocessing on the test problems prior to solution. In Table 1, we show the solution statistics of the LPPEN. The tests were performed on an IBM PS/2-486. The Salford F77 compiler was used. The columns Iter, Reduc, and Refac refer to the number of iterations, number of reductions of the parameter t, and number of refactorizations. The columns f^* and t^* report the objective function value reported on termination and the final value of t, respectively. Finally, we report the solution time in seconds exclusive of input/output under CPU. We stop when

$$|(|c^T x| - |b^T y|)/(1 + |c^T x| + |b^T y|)| \leq 10^{-8}$$

and the largest negative component of the residuals $r = A^T y + c$ is less than $n\|c\|_\infty \epsilon_M$ in absolute value, where ϵ_M denotes the computer unit roundoff. In connection with the initialization, we used $\kappa = \beta = 10^{-1}$ for all the problems. We observe that, although the final value of the penalty parameter

varies in the range between $O(10^{-7})$ and $O(10^{-3})$, the penalty algorithm yields very accurate optimal solutions. This is a very encouraging result, since quadratic penalty function methods have been well known for their potential numerical instability. These experimental findings corroborate our analytical results that the penalty parameter t is decreased to zero in a well-conditioned manner by solving a linear system of equations and taking a step along the solution.

In Table 2 below, we provide a comparison of LPPEN with LPASL1 and LSSOL. We consider only the solution time in CPU seconds. For details on the results of LPASL1 and LSSOL, the reader is directed to Ref. 1. These results were obtained on the same computer under identical compiler settings. The results of Table 2 clearly demonstrate that, on the test set, LPPEN and LPASL1 display very similar numerical performances. Both codes also deliver a solution in times competitive with LSSOL.

4.3. Relation to Interior-Point Methods. Interior point methods solve a weighted least-square system of the form

$$(ADA^T)p=f,$$

where D is diagonal positive-definite matrix and p, f are vectors of appropriate dimensions. Since the numerical values of the entries of D change from one iteration to the next, a numerical refactorization of the matrix ADA^T is performed at each iteration of these methods. However, we do only occasional refactorizations of the matrix $A\bar{W}A^T$. As a preliminary test, we ran some randomly generated dense problems to force CPLEX to use dense linear algebra on a SUN SPARC 4 using the CPLEX Barrier optimizer and

Table 2. Comparison of solution times for LPPEN, LPASL1, and LSSOL on the test set.

Problem name	LPPEN	LPASL1	LSSOL
Afiro	1.09	0.49	0.60
Sc50b	1.59	0.93	3.07
Sc50a	1.70	1.64	2.47
Sc105	6.37	9.78	14.28
Adlittle	5.60	8.13	14.56
Scagr7	17.36	19.06	58.07
Stocfor1	16.20	19.01	15.93
Blend	7.80	7.25	12.96
Sc205	31.26	107.63	122.74
Share2b	12.19	11.86	13.07
Plate	2.69	2.47	9.23

Table 3. Comparison of solution times for LPPEN and CPLEX Barrier optimizer on dense problems on a SUN SPARC 4 (25 MHz).

n, m	LPPEN	CPLEX Barrier
200, 100	5.32	16.95
320, 160	11.52	67.84
400, 200	25.07	140.39
600, 300	117.50	451.79

LPPEN. The results are given above in Table 3. The times are in CPU seconds.

These results clearly show that LPPEN is well suited for dense problems. In the light of the observations, we believe that the penalty method deserves further research.

References

1. MADSEN, K., NIELSEN, H. B., and PINAR, M. Ç., *A New Finite Continuation Algorithm for Linear Programming*, SIAM Journal on Optimization, Vol. 6, pp. 600–616, 1996.
2. MANGASARIAN, O. L., *Normal Solutions of Linear Programs*, Mathematical Programming Study, Vol. 22, pp. 206–216, 1984.
3. MADSEN, K., and NIELSEN, H. B., *A Finite Smoothing Algorithm for Linear l_1-Estimation*, SIAM Journal on Optimization, Vol. 3, pp. 223–235, 1993.
4. HUBER, P. J., *Robust Statistics*, John Wiley, New York, New York, 1981.
5. MADSEN, K., and NIELSEN, H. B., *Finite Algorithms for Robust Linear Regression*, BIT, Vol. 30, pp. 682–699, 1990.
6. BERTSEKAS, D. P., *Necessary and Sufficient Conditions for a Penalty Method to Be Exact*, Mathematical Programming, Vol. 9, pp. 87–99, 1975.
7. MANGASARIAN, O. L., and MEYER, R. R., *Nonlinear Perturbations of Linear Programs*, SIAM Journal on Control and Optimization, Vol. 17, pp. 745–752, 1979.
8. LI, W., PARDALOS, P., and HAN, C. G., *Gauss–Seidel Method for Least-Distance Problems*, Journal of Optimization Theory and Applications, Vol. 75, pp. 487–500, 1992.
9. LI, W., and SWETITS, J., *Linear l_1-Estimator and Huber M-Estimator*, Technical Report, Old Dominion University, Norfolk, Virginia, 1995.
10. MICHELOT, C., and BOUGEARD, M., *Duality Results and Proximal Solutions of the Huber M-Estimator Problem*, Applied Mathematics and Optimization, Vol. 30, pp. 203–221, 1994.

11. GILL, P. E., HAMMARLING, S., MURRAY, W., SAUNDERS, M. A., and WRIGHT, M. H., *User's Guide for LSSOL (Version 1.0): A Fortran Package for Constrained Linear Least Squares and Convex Quadratic Programming*, Report SOL 86-1, Department of Operations Research, Stanford University, Stanford, California, 1986.
12. BARTELS, R. H., *A Penalty Linear Programming Method Using Reduced-Gradient Basis-Exchange Techniques*, Linear Algebra and Its Applications, Vol. 20, pp. 17–32, 1980.
13. CONN, A. R., *Linear Programming via a Nondifferentiable Penalty Function*, SIAM Journal on Numerical Analysis, Vol. 13, pp. 145–154, 1976.
14. CHEBOTAREV, S. P., *Variation of the Penalty Coefficient in Linear Programming Problems*, Automation and Remote Control, Vol. 7, pp. 102–107, 1973.
15. CLARK, D. I., and OSBORNE, M. R., *Finite Algorithms for Huber's M-Estimator*, SIAM Journal on Scientific Computing, Vol. 7, pp. 72–85, 1986.
16. DAX, A., *Linear Programming via Least Squares*, Linear Algebra and Its Applications, Vol. 111, pp. 313–324, 1988.
17. PINAR, M. Ç., *Linear Programming via a Quadratic Penalty Function*, Zeitschrift für Operations Research, Vol. 44, pp. 345–370, 1996.
18. LUENBERGER, D., *Linear and Nonlinear Programming*, Addison-Wesley, Reading, Massachusetts, 1984.
19. NIELSEN, H. B., *AAFAC: A Package of Fortran 77 Subprograms for Solving* $A^TAx = c$, Report NI-90-11, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1990.